scientific reports

OPEN

A hybrid recurrent neural network and optimization framework for intelligent mobile robot navigation in smart manufacturing

K. Radha[1]✉ & S. Karthikeyan[2]

In the third dimension prospective of production industries taking on smart manufacturing principles, the integration of automation and digitalization revolutionizes conventional processes, unlocking heightened productivity and operational efficiency. This endeavour implicates coordinating unified interactions among machines and human operators, capitalizing on their unique strengths and capabilities. In this study, a multi-objective optimal navigation system tailored for mobile robots operating in dynamic surroundings, leveraging hybrid optimization algorithms. Primarily, introduce the modified animal's migration optimization (MAMO) algorithm, which measures obstacle state data essential for dynamic obstacles. This facilitates proactive collision avoidance, thus minimizing unnecessary disruptions. Consequently, deep features are extracted from all feasible paths spanning the target region connecting the origin and destination points. These path features are then subjected to the chaos locust search (CLS) algorithm, which determines multiple paths to consider. In addition, the hypercube search with recurrent neural network (HS-RNN) is employed to locate the optimal path while removing redundant alternatives among the multiple choices, thereby refining the path planning process. Simulation outcomes highlight the better performance of the proposed system in optimal path generation compared to alternative approaches, validating its efficacy in dynamic environments.

Keywords Navigation system, Mobile robot, Smart manufacturing, Production industries, Obstacle state information

Abbreviations

NP	The total number of animal agents in the population
D	The number of variables in the optimization problem
$X_{i,G}$	Current State position of the obstacle (i) at generation (G)
Xneighborhood,G	State of randomly selected Neighboring obstacle at generation (G)
$X_{i,G+1}$	New State position of the obstacle through the neighborhood
Δ	Gaussian random parameter between 0 and 1
rand (0,1)	A uniformly distributed random number between 0 and 1
Pa	A threshold probability used in the population update process. An agent is updated if a random number is greater than Pa
r1, r2	Randomly generated indices within the population [1, NP], used in the position update equation
L = {l1,l2,…lN}	The population of N locust agents (path solutions)
Lk = {lk1,lk2,…lkN}	The position (a candidate path) of the ith locust agent at iteration k
l_i^k	Position of agent (i) at iteration
Lk	The entire population of path agents at iteration
S_i^k	Total social force acting on agent (i) at iteration
S_{ij}^k	Pairwise social force between agents (i) and (j)
$\rho\left(l_i^k, l_j^k\right)$	Dominance value of agent (i) relative to agent (j)
r_{ij}^K	Distance between agents (i) and (j)

[1]Department of Computer Science and Engineering, Dr Mahalingam College of Engineering and Technology, Pollachi, Tamilnadu, India. [2]Department of Mechanical Engineering, Christian College of Engineering and Technology, Oddanchatram, Tamilnadu, India. ✉email: radhacse.dpm@gmail.com

$s\left(r_{ij}^{K}\right)$	Social factor function (attraction/repulsion)
F, L	Parameters controlling the magnitude and forces
pk	Behavior probability (Solitary vs. social phase) k
Bk	Subset of q best solutions at iteration k
$A\left(b_j^K\right)$	The attractiveness of solution
f_{best}, f_{worst}	The best and worst fitness within the set B^k.
$\in$	A very small constant to stop division by zero
$b^k{}_r$	A guide solution randomly selected from B^k
C	Chaotic variable from the logistic map
X [ ]	Random (n, m) are formed based on the parameters dimension (m), radius (R), lower and upper bounds (LB, UB), and population size (n)
LB, UB	Lower and Upper bound of search space
R	Radii of the hybercube (R = UB –LB)
X_c	Center of hybercube
X_{best}	The best solution found in the current hybercube
S	Converge factor of the hybercube
d_{nn}	Normalized distance matrix

Smart manufacturing[1] refers to the efficient utilization of labor, materials, and energy to produce tailored, high-quality products with a technology-driven approach, ensuring timely delivery. These traits collectively contribute to the resilience and sustainability of manufacturing by enhancing resource and energy management[2,3]. Autonomous navigation is a pivotal capability for mobile robots, reducing their dependence on human intervention[4]. Path planning entails decisive the most efficient sequence of achievement for a robot to move from its present state to a desired one, with these states representing the initial position and the goal, respectively[5]. The swift evolution of mobile robots not only enhances daily conveniences, exemplified by technologies like sweeping robots, but also plays a vital role in substituting workers in perilous industries such as mining and aerospace[6]. Within intelligent building systems[7], inspection robots exhibit the capability to strategically plan the shortest path to navigate a path from danger in critical situations like fires. Path planning algorithms[8,9] usually try to find the best path or acceptable supposition. For instance, the ideal path could be one that restricts the time required, an essential figure missions like request and-rescue errands, where brief assist with canning unfathomably critical issue[10]. These constraints may stem from the robot's limitations in adapting to specific terrains[11,12]. Conversely, dynamic path planning[13] is more intricate, capable of adapting to real-time changes in the environment to plan a route for a moving robot. Robot path preparation methods fall into two main classes such as conventional and modern systems[14]. Classic techniques incorporate cell decay, likely field, sub-target, and examining based strategies[15,16]. Heuristic algorithms comprise neural networks[17], fuzzy logic[18], normal heuristic techniques[19], and half-breed calculations. While exemplary strategies might confront difficulties in additional working on the proficiency of path search and improvement, prompting a progressive decrease in usage, heuristic techniques have acquired notoriety for their viable worldwide streamlining capacities and parallelism[20].

The literature review on navigation system for mobile robot for smart manufacturing focused the research gaps in existing literatures, particularly on the type of technique used for path finding. Luo et al.,[21] planned an improved ant colony algorithm (IAC) to resolve issues connected with neighborhood enhancement, unfortunate assembly, and low pursuit proficiency in portable robot average path length (PL) and delay finding. Maoudj et al.,[22] introduced an effective Q-learning to address difficulties, guaranteeing the deduction of an ideal crash free path in negligible computational time and safety. Ajeil et al.,[23] introduced a hybridized particle swarm optimization-modified frequency bat calculation, expecting to limit PL/distance while sticking to path perfection standards. Chen et al.,[24] have introduced the bio-spurred organizing computations for the effect free path orchestrating of robots in strong circumstances, expressly without a hint of prior information. The padding mean mind dynamic model is composed association with relationship among abutting neurons, planned to work with the spread of nerve main thrusts like waves without coupling influences. Li et al.,[25] have introduced an enhanced rapidly-exploring random tree (PQ-RRT) algorithm, showing culmination, asymptotic optimality, and quick convergence rate to the ideal arrangement. RRT keeps up with a similar computational intricacy. Albeit static path arranging is transcendently investigated at the on-going phase, certifiable conditions are dynamic. Miao et al.,[26] have presented an enhanced adaptive ant colony algorithm intended for enhancing the path arranging and length findings of indoor portable robots. Song et al.,[27] have devised a strategy for addressing the planning challenge of achieving a smooth robot path with accuracy and F-measure. This includes using a constant serious level Bezier bend related to an improved particle swarm optimization (IPSO) calculation. Instead of associating various low-degree Bezier bend fragments, the persistent serious level Bezier bend is utilized to meet the rules for smooth path arranging. Lyu et al.,[28] have introduced a graph-based system aimed at optimizing robot average paths and computational time, with the focal point being the application of the Floyd algorithm to dynamically allocate masses to various paths. Zhang et al.,[29] have presented extrapolative path planning designed for robots navigating dynamic surroundings, utilizing the RRT. It speeds up the time to revaluating framework, restricting the path cost, diminishing the likelihood of accidents, and finally working on the idea of the re-examined path. Nayab Zafar et al.,[30] have introduced a navigation control methodology utilizing a hybrid grey wolf's optimization and the artificial potential field technique for timely path preparation of mobile robots. Existing path planning methods struggle with adaptability and efficiency in dynamic, complex environments due to limitations in optimality, continuity, and computational speed. Yang et al.,[31] have introduced a unique robot path preparation that facilitates a predominant improved A* algorithm with better strong window technique. Zhong et al.,[32] have utilized a classic A and safe A for mobile robot in large-scale environment. Raj et al.,[33] presented the reinforcement learning technique for intelligent mobile robot navigation problem.

Many existing classical algorithms often fail to provide globally optimal paths in dynamic environments. Most research papers assume static environments and real-time path planning that adapts to moving obstacles in smart manufacturing settings is still a major challenge. The integration of hybrid optimization approaches remains underexplored. The main contribution of this research paper is intended to improve the dynamic path planning by combining the MAMO, CLS and HS-RNN, to overcome the drawbacks of conventional tools such as poor exploration and local optima entrapment. In the proposed approach, MAMO is applied to measure obstacle states, and CLS is employed to explore the various paths by chaotic theory and a HS-RNN is used to determine the optimal path. Hence, the primary objectives of this work is to efficiently design an intelligent multi-path decision-making navigation scheme by leveraging optimal feature extraction from all potential paths between the source and destination in static and dynamic scenarios. Also, the proposed navigation scheme has been validated with existing solutions to ensure suitability for smart manufacturing in production industries.

Problem environment

In this study, the environment modeling relies on the grid technique, a technique that partitions the two-dimensional workstation of mobile robots into a uniform grid structure. The illustration in Fig. 1 demonstrates the division of the entire workspace into a grid map, where each grid is allocated a unique amount. Initially, information is gathered from the grid map, utilizing the discrete nature of the environment.

The experimental environment is defined as a (20×20), (40×40), (50×50), (80×80) and (100×100) grid maps, where each cell represents a possible position for the mobile agent. Black cells denote obstacles, and white cells correspond to free navigable spaces. The agent starts at the lower-left corner and aims to reach the goal position located at the upper-right corner. The grid includes both static and dynamic obstacles to emulate a real-world navigation scenario. Static obstacles remain fixed throughout the simulation and form the baseline environment for path generation. In contrast, dynamic obstacles change their positions over time according to predefined movement rules. During navigation, the MAMO algorithm computes the current obstacle states at each time step by updating the availability matrix. When a dynamic obstacle moves into a previously free cell, its corresponding entry in is updated from 0 (free) to 1 (occupied/obstacle) and vice versa if the obstacle moves away. For example, consider a navigation scenario on a 5×5 grid where coordinates (x, y) represent positions, with x-values from 1 to 5 spanning left to right and y-values from 1 to 5 spanning bottom to top. The agent's starting point is S = (1, 1), and the target destination is G = (5,5). Agent mobility is continuous, meaning positions are not confined to discrete grid points; however, for the purpose of obstacle detection, these continuous coordinates are rounded to the nearest integer grid cell. The environment includes predefined static obstacles at cells (2,5), (4,4), and (3,2), forming an initial binary obstacle map. Additionally, a dynamic obstacle, unknown to the agents initially, starts at (2,3) at time $t=0$ and transitions to (2,4) at $t=1$. Agents can only identify obstacles, whether static or dynamic, by physically occupying the corresponding cell during movement. This simulation utilizes in Eq. (1)[34] with a parameter $\delta=0.5$ and in Eq. (3)[34] with $\rho=0.6$. Over the course of two MAMO iterations (from generation 0 to generation 2), an occupancy detection matrix is updated. This matrix begins with the static obstacles already marked, and a cell is flagged as detected if any agent lands on it during iteration.

Initial state

5	0	1	0	0	G
4	0	0	0	1	0
3	0	0	0	0	0
2	0	0	1	0	0
1	S	0	0	0	0
	1	2	3	4	5

The MAMO begins with three agents positioned near (1,1), (3,1), and (2,4). In the first iteration (G = 0→1), migration guides the agents to new sensing cells at (2,2), (4,2), and (2,4), but none encounter the dynamic obstacle D at (2,3). A subsequent population update replaces the worst-performing agent, repositioning it to (2,4). In the next iteration (G = 1→2), the dynamic obstacle moves to (2,4). Agent migration now leads two agents to land on cell (2,4), resulting in the successful detection of the dynamic obstacle. This discovery is then integrated into the observed obstacle matrix, updating the system's knowledge of the environment.

5	0	1	0	0	G
4	0	1	0	1	0
3	0	0	0	0	0
2	0	0	1	0	0
1	S	0	0	0	0
	1	2	3	4	5

(2,4) = 1 because two agents confirmed observation by landing there.

The CLS is employed to select multiple paths based on the extracted features. CLS introduces chaos to enhance exploration, considering various alternatives for navigation. For example, the iteration begins with the solitary phase, where pairwise social forces are computed for the three path locusts. For the worst locust l_3 (P_3), forces from the better-ranked locusts are calculated using the defined s(r) function and dominance rule ρ. The sum of these forces S_3 is a vector, which is added to l_3 to update its position. In the subsequent social phase, the best set B is formed from the top two paths, P_1 and P_2. Their attractiveness values are calculated to be approximately 1.0. For the updated worst locust, the roulette selection probabilities for P_1 and P_2, respectively. P_1 is selected as the attractor. Finally, the social update rule with a random value of 0 to 1 is applied, moving the worst locust to a new position, very close to the best centroid P_1. Further illustrated, the PL[38], measured as the total number of moves, was identical for all three candidate paths at 8 units. However, the paths differed significantly in smoothness, quantified by their cumulative turning angle (CTA)[31]. Paths P_1 and P_2, which consisted of straight segments with only one direction change, had a low CTA of 90°. In contrast, Path P3's zig-zag pattern resulted in five turns, yielding a cumulative turning angle of 450° five times higher. While the average turning angle (ATA)[32] was 90° for all paths, as each individual turn was a right angle, the cumulative metric effectively captured P_3's tortuosity, making it a key differentiator for path quality where smoothness impacts efficiency. Finally, the HS-RNN model is to identify the optimum path from the alternatives generated by CLS. The model discerns between optimal and redundant paths, refining the path selection process. The MAMO + CLS + HS-RNN system was implemented in MATLAB programming simulation on an Intel core I5-9400 CPU and 16 GB RAM platform. Key parameters included: MAMO with 20 agents and a migration rate (δ = 0.5). The CLS algorithm used a population of 50, with *F* (social force magnitude) = 1.5, *L* (social interaction range) = 1.0, and *q* (number of best solutions for attraction) = 10. The HS-RNN performed multi-criteria optimization using PL, a turning penalty (weight = 0.01), and safety. Multiple independent runs validated the results.

Proposed methodology

In this section, it describe the multi-objective optimal navigation system integrates grid-based environment model.

Compute obstacle state information

Obstacle state information refers to a dynamic assessment of the current state or characteristics of obstacles within the environment. The need to compute obstacle state information arises from the necessity to enhance the ability to direct through its environment effectively. The modified animal's migration optimization (MAMO) algorithm is a nature-inspired optimization that draws stimulus from the collective performance of animals[34]. During the migration procedure, the algorithm pretends collections of animals moving from an existing state. In the process of informing the population, the algorithm pretends the probabilistic renewal of animals. If the index of an animal is *i*, its neighborhood consists of animals having indices are $i-2, i-1, i, i+1, i+2$, and if the index is 1, then the neighborhood indices $NP-1$, NP, 1, 2, 3, etc. Once the locality topology is created, randomly choose a neighbor and appraise the individual's location rendering to that neighbor, using Eq. (1)[34].

$$X_i, G+1 = X_{i,G} + \delta\ (Xneiborhood,\ G - X_{i,G}) \tag{1}$$

The new position $X_{i,\,G+1}$ is calculated using the current position $X_{i,\,G}$, a neighborhood position $X_{\text{neighborhood},\,G}$, a Gaussian random parameter δ, and the vector difference between two other randomly selected individuals (r_1, r_2). The better position between the new and current one is selected using Eq. (2)[34].

$$X_i = \begin{cases} X_i\ if\, f\,(X_{i,G}\,)\ \ is\, betterthan\, f\,(X_{i,G+1}\,) \\ X_{i,G+1}\ \ \ otherwise \end{cases} \tag{2}$$

In the MAMO, the best individual defines the G^{th} cycle living region. As resources diminish, few animals begin to migrate, developing the $G+1$th cycle living region. This continuous shrinking of the living region makes individuals converge toward the best string/solution, improving convergence speed and accuracy.

The boundary of the living region is recognized as in Eq. (3)[34].

$$Low = X_{best} - R,\ \ up = X_{best} + R,\ \ R = \rho\, R \tag{3}$$

Here, X_{best} denotes the leader animal; low and up are the living region bounds; *R* is the radius; is the shrinkage coefficient; and ∈ (0, 1), low, up and *R* are all 1×*D* array vectors. Initially, *R* depends on the search region size and larger *R* improves exploration, while smaller *R* enhances exploitation during cycles.

The MAMO contains: (a) animals living in a defined region, (b) migrating as resources deplete, (c) population updating, and (d) settling in a new region. So, it's covering living, migration, and updating processes.

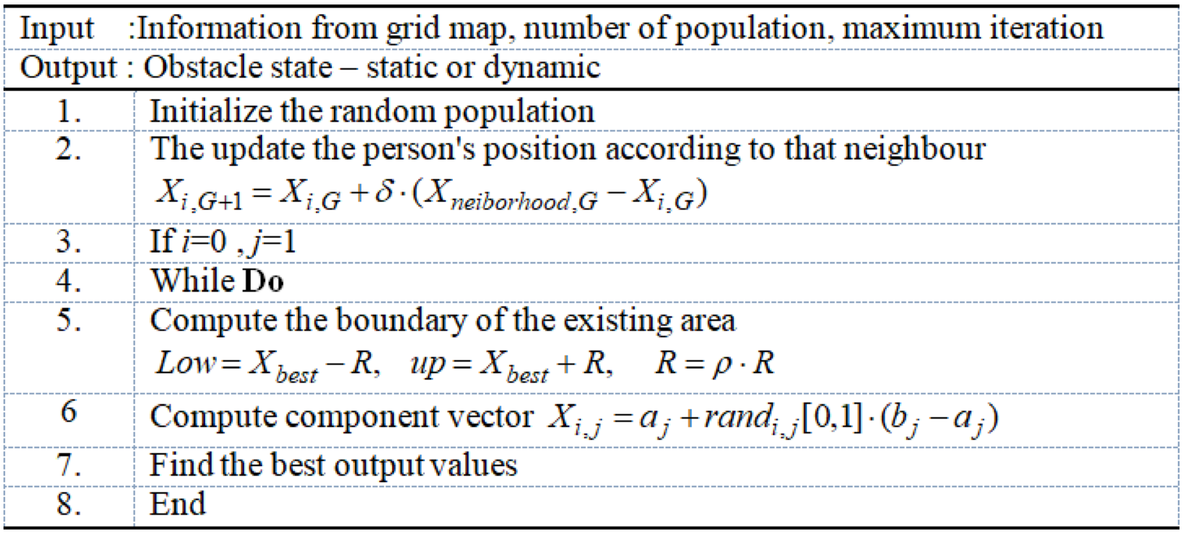
In the initialization, the MAMO starts by initialize a set of animal locations $X_1, X_2, X_3, \ldots X_{NP}$, every location X_i D-dimensional vector with components uniformly distributed among lower bound a_j and upper bound b_j. So the j^{th} component of the i^{th} vector as Eq. (4)[34] and (5)[34].

$$X_{i,j} = a_j + rand_{i,j}\,[0,\ 1]\ (b_j - a_j) \tag{4}$$

$$i = 1,\quad NP,\ j = 1,\quad D \tag{5}$$

where, [0,1] *i j* rand is a uniformly distribution.

In MAMO, migration occurs as resources deplete or conditions change, updating grid map and locations based on neighbors. During new population updating, replaced individuals maintain a fixed population size, allowing computing the obstacle state and ensuring continuity of the algorithm. The working process of obstacle state information computation using MAMO is explained in Algorithm 1.



```
Input  :Information from grid map, number of population, maximum iteration
Output : Obstacle state – static or dynamic
1.  Initialize the random population
2.  The update the person's position according to that neighbour
    $X_{i,G+1} = X_{i,G} + \delta \cdot (X_{neiborhood,G} - X_{i,G})$
3.  If $i$=0 ,$j$=1
4.  While Do
5.  Compute the boundary of the existing area
    $Low = X_{best} - R,\quad up = X_{best} + R,\quad R = \rho \cdot R$
6   Compute component vector $X_{i,j} = a_j + rand_{i,j}[0,1] \cdot (b_j - a_j)$
7.  Find the best output values
8.  End
```

Algorithm 1. Obstacle state information computation usign MAMo.

Feature extraction and multiple path selection

Feature extraction is the procedure of capturing and representing relevant features from raw data. In the context, feature extraction is functional to paths within the grid map to capture essential information that aids in the navigation process. The CLS algorithm is used for selecting multiple paths which enhance the exploration, considering various alternatives for navigation. CLS is a population-based global optimization method inspired by the collective movement of desert locust swarms[35]. In the CLS algorithm, search agents are represented by an L = {l_1,l_2,…l_N} set of N individual locusts that interact with each other as they traverse the n-dimensional solution space. Every individual condition l_i = {$l_{i,1}$,$l_{i,2}$,…$l_{i,n}$} is constrained in bounded region (S = { $x \in R^n$ | $lb_d \leq x_d \leq ub_d$}), with $x = [x_1, x_2, x_3, x_d]$, lower lb_d, upper ub_d bounds at the d-dimensional and characterizes applicant's string to a particular problem. Like other swarm-based methods, CLS proceeds by cycles where agents alter their locations across generations. These updates are guided by operators modeled on the two behavioral phases of locusts such as solitary and social phase. During the solitary phase, individuals move to different locations in search of food sources, while avoiding mating with other potential partners. Hence, for any cycles 'K' is the total attractive and repulsion force by a particular individual 'i', as in Eq. (6)[35].

$$S_i^k = \sum_{\substack{j=1 \\ j \neq 1}}^{N} S_{ij}^k \tag{6}$$

where S_{ij}^k denotes the pairwise attraction among separate "i" and some other individuals "j" and is assumed by Eq. (7)[35].

$$S_{ij}^k = \rho\left(l_i^k, l_j^k\right)\, s\left(r_{ij}^k\right) d_{ij} + rand\,(1, -1) \tag{7}$$

where the operative$\rho\left(l_i^k, l_j^k\right)$ is known as supremacy value amongl_i^k and l_j^k the worth$s\left(r_{ij}^k\right)$characterizes the so named social factor, where $r_{ij}^k = \left\|l_i^k, l_j^k\right\|$ ‖ denotes the Euclidian distance among the elements "i" and "j". Therefore, $d_{ij} = \left\|l_i^k, l_j^k\right\| / r_{ij}^k$attitudes for the unit course cement from l_i^k to 1 l_j^k, though and (1, − 1) is a random vector whose fundamentals are haggard from the unchanging distribution of [− 1, 1]. The value $s\left(r_{ij}^k\right)$ is assumed by Eq. (8)[35].

$$s\left(r_{ij}^k\right) = Fe^{-r_{ij}^k/L} - e^{-r_{ij}^k} \tag{8}$$

where the limitations f and l represent the attraction greatness and measurement scale, individually. To apply operator$\rho\left(l_i^k, l_j^k\right)$ and presumed that each individual $l_i^k \in l^k = \left\{l_1^K, l_2^K, l_N^K\right\}$ is ordered with number from 0

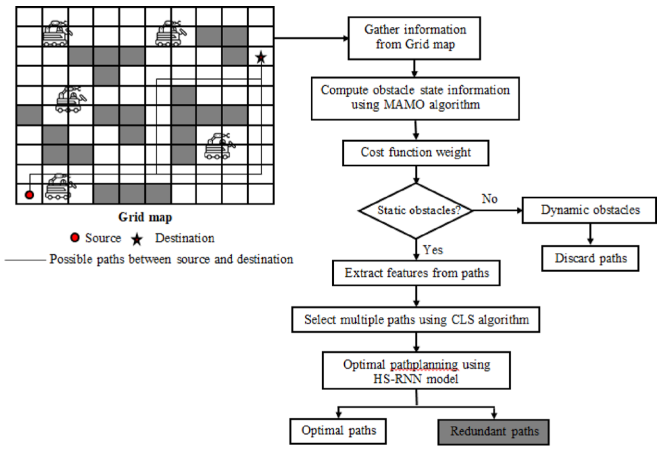

Grid map
● Source ★ Destination
Possible paths between source and destination
Gather information from Grid map
Compute obstacle state information using MAMO algorithm
Cost function weight
Static obstacles?
No
Dynamic obstacles
Discard paths
Yes
Extract features from paths
Select multiple paths using CLS algorithm
Optimal pathplanning using HS-RNN model
Optimal paths
Redundant paths

Fig. 1. Overall system architecture of proposed multi-objective optimal navigation system.

(Best) to $N-1$ (Worst) contingent on their individual fitness worth. Then, the dominance worth may be assumed as in Eq. (9)[35]:

$$\rho\left(l_i^K,\, l_j^K\right) = \begin{cases} e^{-\left(rank\left(l_i^K\right)/N\right)} \; if \; rank\left(l_i^K\right) \le \left(l_j^K\right) \\ e^{-\left(rank\left(l_i^K\right)/N\right)} \; if \; rank\left(l_i^K\right) > \left(l_j^K\right) \end{cases} \tag{9}$$

As effect of the inspiration of the total social forcess_i^k, each person shows a certain propensity to move towards other memberships of the "*i*" population. In such cases, the new location resulting from individual "*i*" can bess_i^k expressed as in Eq. (10)[35].

$$l_i^* = l_i^k + S_i^k \tag{10}$$

Applying solitary phase movement operators to every individual separately leads $l_i^* \in l^k$to refine the best candidate strings $L = \{l_1, l_2, \ldots l_N\}$that represent every individual's location as an outcome of the inspiration used by all other members of the swarm.

In the social phase, the aim is to improve the best candidate strings ($L = \{l_1, l_2, \ldots l_N\}$), which are attained by the solitary phase movement operator. From this set, a subset ($B = b_1, \ldots . b_q$) containing the best strings (q) is selected. For each string ($l_i \in B$), a group of (h) random solutions ($M^i = \left\{m_1^i, \quad m_i^h\right\}$) is created within its respective subspace ($C_i \in S$), where the boundaries of are predefined, as in Eq. (11)[35] and Eq. (12)[35].

$$C_{i,n}^{lower} = b_{i,n} - r \tag{11}$$

$$C_{i,n}^{upper} = b_{i,n} + r \tag{12}$$

where$C_{i,n}^{lower}$ and $C_{i,n}^{upper}$characterize the lower and upper bounds of every sub region c_i at the *n*-th dimension, correspondingly, while $b_{i,n}$attitudes for the n^{th} element from string b_i, as in Eq. (13)[35].

$$r = \frac{\sum_{n=1}^{d}\left(b_n^{upper}\right) - \left(b_n^{lower}\right)}{d}\beta \tag{13}$$

whereb_n^{lower} and b_n^{upper} denoting the b represents the lower and higher bounds, and d represents the total amount of decision variables. Also, $\beta \in [0,\ 1] \in [0,\ 1]$ characterizes a scaling factor that changes the scale C_i. Finally, the best string $l_i \in N$ and its corresponding i from the random strings ($\left(m_1^i,\, m_2^i,\, m_3^i\right)$) are assigned to a unique location "*i*" in the next iteration "$k+1$", expressed in Eq. (14)[35].

$$l_i^{k+1} = best\ \left(m_1^i,\, m_2^i, \quad m_h^i\right) \tag{14}$$

Any solution not grouped in the best solution l_i^*set **B** is not eligible for exclusion by the social status director. Therefore, the last location update practical to each individual in the entire swarm by '*i*' can be summarized as in Eq. (15)[35]:

$$l_i^{k+1} = \begin{cases} best\ \left(l_i^*,\, m_1^i,\, m_2^i, \quad m_h^i\right)\ if\ l_i^* \varepsilon\, B \\ l_i^* \qquad\qquad\qquad\quad l_i^* \notin\ B \end{cases} \tag{15}$$

The CLS combines global and local search by solitary and social phases but suffers from maximum computational imbalance among exploration and exploitation. It introduces a probabilistic pattern that applies either phase selectively and adjusts the social operator with a probabilistic attraction, improving performance and efficiency. The solitary and social phases are based on cycles. Early steps emphasize solitary exploration, while later steps favor social exploitation. The probability of social behaviors increases in the search progresses, guiding the choice of the phase at every cycle. Underneath such conditions, at a piece repetition "*k*", the behavior phase $P\left(l^k\right)$functional to the populace $l^k$is selected as shadows in Eq. (16)[35].

$$P\left(l^k\right)=\begin{cases} solitary\ if\ rand\ \le p^k \\ solitary\ if\, rand\ >\ p^k \end{cases} \tag{16}$$

A random number between 0 and 1 is compared against a performance probability, p^k, calculated using Eq. (17)[35].

$$p^k=1-\frac{K}{itern} \tag{17}$$

The number of cycles measured in the search process is indicated as itern. For this modified social operator, the social phase leads each individual toward hopeful strings instead of relying on multiple local valuations, thereby minimizing computational time. A subset of the best strings ($B^k = b^k{}_1,\ldots b^k{}_q$) is chosen that includes the q best strings from the total set $L^k = \{l^k{}_1, l^k{}_2, \ldots l^k{}_N\}$, and every individual $l^k{}_i$ moves toward a randomly chosen $b^k{}_j$ with a probability affected by both its quality and distance, as in Eq. (18)[35].

$$p^K_{1_i,b_j}=\frac{A\left(b^k_j\right)e^{-\left\|l^k_i-b^k_j\right|}}{\sum_{n=1}^{q}A\left(b^k_n\right)e^{-\left\|l^k_i-b^k_n\right.}} \tag{18}$$

where $\left\|l^k_i-b^k_j\right\|$denotes the Euclidian distance among the separate "*i*" (l^k_i) and associate "j" from the set of greatest strings$B^k\ \ \left(b^k_j\right)$, while $A\left(b^k_j\right)$stands for the attractiveness of string b^k_j as mentioned by Eq. (19)[35].

$$A\left(b^k_j\right)=\frac{f\left(b^k_j\right)-f_{worst}\left(B^k\right)}{f_{best}\left(B^k\right)-f_{worst}\left(B^k\right)+\varepsilon} \tag{19}$$

It represents$f\ \left(b^k_j\right)$ the relative fitness value b^k_j (rank) $f_{best}\ \left(B^k\right)\ f_{worst}\ \left(B^k\right)$ stand the best and worst fitness values from the members of the best string sets. The social phase characteristics, every individual $l^k{}_i$ within the swarm population L^k updates their location as in Eq. (20)[35]:

$$l^{k+1}_i=l^k_i+2\left(b^k_r-l^k_i\right)\ rand \tag{20}$$

where $b^k{}_r$ (by r ∈ [1,. . ., *q*]) is a randomly selected explanation $b^k{}_j \in B^k$, with respect to their particular prospectsP^k_{ij} (qualified to $l^k{}_i$ and $b^k{}_j$), while rand stands for a random haggard from within the consistently dispersed interval [0, 1]. The working process of feature extraction and multiple path selection using CLS is explained in Algorithm 2. The extracted path features are then subjected to the CLS algorithm, which produces a set of multiple feasible paths rather than a single solution. This step ensures that the diversity of potential routes is preserved.

```
Input   :Static and dynamic obstacle information
Output  :Multiple paths
1.   Initialize the random population
2.   Define the value $s(r_{ij}^K)$ is given by $s(r_{ij}^K) = Fe^{-r_{ij}^K/L} - e^{-r_{ij}^K}$
3.   If $i$=0 , $j$=1
4.   While Do
5.   Compute individual "$i$" as a result of $S_i^K$ expressed as
     $l_i^* = l_i^K + S_i^K$
6    Compute the whole swarm $l_i^{K+1} = \begin{cases} best(l_i^*, m_1^i, m_2^i, \dots m_h^i) & if\ l_i^* \varepsilon B \\ l_i^* & if\ l_i^* \notin B \end{cases}$
7.   If not discard then
8.   Compute distance separating both individuals ($l_i^k$ and $b_i^k$)
     $P_{1_i,b_j}^K = \frac{A(b_j^K)e^{-\|l_i^K - b_j^K|}}{\sum_{n=1}^{q} A(b_n^K)e^{-\|l_i^K - b_n^K|}}$
9.   Define the swarm population $L^k$ updates their position
     $l_i^{K+1} = l_i^K + 2(b_r^K - l_i^K) \cdot rand$
10.  Store all feasible paths generated in the current iteration into Path_set.
11.  End while loop
12.  Return the collection of all feasible paths as the final output
```

Algorithm 2. Feature extraction and multiple path selection usig CLS.

Optimal path planning

After the assortment of multiple paths, the optimum path planning process involves identifying the most efficient and effective path while eliminating redundant or suboptimal alternatives. In this context, the HS-RNN is employed to enhance the optimization of the planning process, and equations are taken from Tunay and Rahib[36] whose advantages are utilized in different applications[37]. During initialization, candidate strings (X [search region] = random (n, m) are formed based on the parameters dimension (m), radius (R), lower and upper bounds (LB, UB), and population size (n). The hypercube is defined by its radius (R) and center (X_c), given by Eqs. (22) and (23). Within the defined search region, points x_{ij} ($i = 1,.,n; j = 1,.,m$) are created in Eq. (21), and their corresponding objectives f_{ij} (elements of F) are attained. Thereafter, the best string matrices X_{best} and F_{best} ($n\times1$) are observed, creating the initial population for subsequent cycles. The situation of the X_{best} using local search and following measures$X_{best}^{new} = X_{best} + \rho\Delta F$, where F is the objective meaning and$0 \leq \rho \leq 1$.

Lower bound (LB) and upper bound (UB) to scale the strings x_{ij} in Eq. (21)[36]

$$X_{i,j} = LB + X_{i,j}\,(UB - LB) \tag{21}$$

Determine radii (R) of the hybercube expressed in Eq. (22)[36]:

$$R = UB - LB \tag{22}$$

Center of hybercube are gotten in Eq. (23)[36].

$$X_c = \frac{R}{2} \tag{23}$$

$$X_{new\_center} = \frac{(X_c + X_{best})}{2} \tag{24}$$

$$R_{new} = R * S \tag{25}$$

In the initialization process, next cycle, the X_{best} is used to compute the hypercube center. This process is gathered through calculating the center and mean of the last location (X_c) and the last best X_{best}. In this displacement shrink process, the algorithm computes the new hypercube center and estimates the objective. The next hypercube center is attained through averaging the current hypercube center and the best string (X_{best}). Relationally, it is given in Eq. (24)[36] and Eq. (25)[36].

The convergence factor S is used to calculate new R (R_{new}) based on an old one, R. This operation gradually minimizes the hypercube size and, subsequently, the search region, a stage called shrink. As the hypercube deals, the point density (population) increases. The movement of the best string is governed by contraction, which is higher for lesser displacements, ensuring quicker convergence while escaping local minima. At every cycle, new points are produced, and their functions are computed. Based on these outcomes, the hypercube size is updated, smaller with every stage, leading to a denser search region and fast convergence toward the optimum path. The algorithm thus explores a sequence from the current position, where the displacement ranges are defined, as in Eqs. (26–29)[36]:

Normalizedx_{ij}:

$$x_{ij}^{n} = \frac{(x_{ij} - X_c)}{R} \tag{26}$$

Normalizedx_{best}:

$$x_{best}^{n} = \frac{(x_{ij} - X_c)}{R} \tag{27}$$

Normalized distance d_n:

$$d_n = \frac{\left[sum\left(x_{ij}^{n} - X_{best}^{n}\right)^2\right]^{1/2}}{R} \tag{28}$$

Re-Normalized distance:

$$d_{nn} = \frac{d_n}{\sqrt{m}} \tag{29}$$

At every cycle, element of x is first divided by its respective interval, converting the displacement into unit-scaled points as given in Eqs. (26)[36] and (27)[36]. These normalized values are then further divided through the diagonal length P_{ij}^{k} as given in Eqs. (28)[36] and (29)[36]. This normalization increases the contraction rate of the hypercube, resulting in progressively smaller movements toward the optimum solution.

In this searching area phase, the distances between the new and previous optimal solutions are computed using Eqs. (26–29)[36]. The process also employs the defined interval for renormalization to adjust the search region of x dynamically throughout$\sqrt{m}$, ensuring efficient exploration around the updated optimum.

When the movement x satisfies the given condition, the convergence factor (S) is calculated and updated at each cycle using Eq. (30)[36].

$$S = 1 - 0.2e - 3d_{nn} \tag{30}$$

Here, d_{nn} represents the normalized distance attained from Eq. (29)[36], found from the average of the 2 latest best x values. This mechanism guides the population toward the minimum efficiently by continuously reducing the hypercube area after each cycle. The entire process is recurrent until the decisions are gratified. The algorithm 3 depicts the function of optimal path planning using HS-RNN.

```
Input  :Number of selected paths, maximum iteration and threshold condition
Output :Optimal best path
1.  Initialize the random population
2.  Compute sequence vectors $LB$ and $UB$
3.  If i=0 , j=1
4.  While Do
5.  Define central standards $X_c = \frac{R}{2}$
6.  Perform normalization of $x_{ij}$ and $x_{best}$ normalization
7.  Compute regularized distance $d_n = \frac{\left[sum\left(x_{ij}^n - X_{best}^n\right)^2\right]^{1/2}}{R}$
8.  Compute factor of S and updated with each iteration
        $S = 1 - 0.2e^{-3d_{nn}}$
9.  Compute hypercube parameters are modernized by $R_{new} = R * S$
10. End if
11. Update the final value
12. End
```

Algorithm 3. Optimal path planning using HS-RNN tenique.

Result and discussions

In this section, the validation of proposed MAMO + CLS + HS-RNN system compared with existing methods in both static and dynamic obstacle environments. To assess the static and dynamic path planning performance of method, conduct comparisons with established systems, including classic A*, safe A*[32], and improved A*[31].

Simulation setup

To facilitate performance analysis, this study employs specific surroundings for mobile robots and the conservation map. The map is defined with a designated starting and ending point, and some areas may contain unknown obstacles. Dynamic obstacles within the map move in a straight line at a consistent speed, although their moving direction and position remain unknown. Equipped with sensors, the mobile robot can perceive information within aim perfect range, including the position and speed of obstacles. The mobile robot itself maintains a continuous speed and is capable of movement in all directions. Its maximum velocity, angular velocity, velocity resolution, angular velocity resolution, acceleration and angular acceleration of 1 m/s, 20°/s,

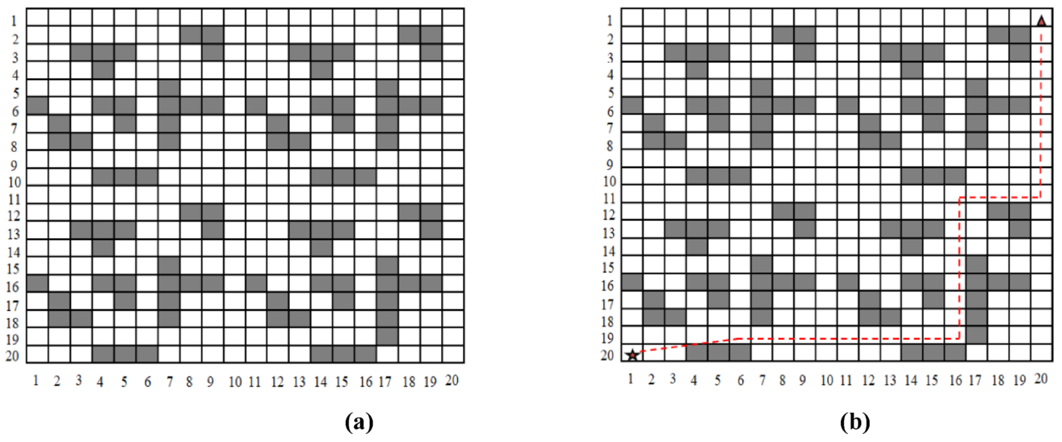


Fig. 2. (**a**) Input grid map (Static obstacles) (**b**) Output optimal path (Static obstacles).

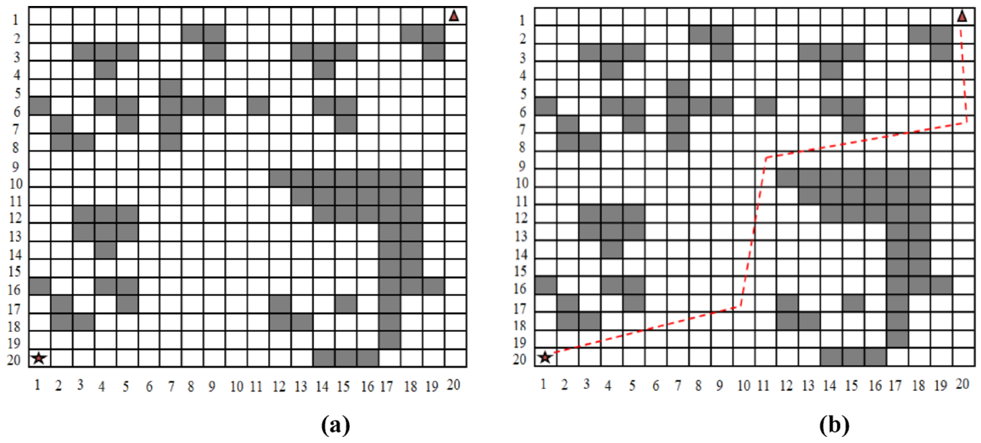


Fig. 3. (**a**) Input grid map (Dynamic obstacles) (**b**) Output optimal path (Dynamic obstacles).

Path planning system	20 × 20	50 × 50	100 × 100	200 × 200	20 × 20	50 × 50	100 × 100	200 × 200
	CTAs (°)				ATAs (°)			
Classic A*[31,32]	488.9	–	–	309.4	32.56	–	–	61.9
SafeA*[31,32]	128.92	–	–	273.4	25.78	–	–	54.7
Improved A*[31]	85.41	–	–	–	14.24	–	–	–
MAMO + CLS + HS-RNN	52.32	55.64	58.04	229.2	5.62	6.34	8.96	45.8
PL					PPT			
Classic A*[31,32]	28.70	82.80	169.70	1051.4	22	26	155	–
SafeA*[31,32]	25.60	67.20	77.30	1065.8	18	22	85	–
Improved A*[31]	26.10	61.90	74.80	–	10	16	42	–
MAMO + CLS + HS-RNN	**22.15**	**31.25**	**41.23**	**1038.7**	6	8	19	–

Table 1. Results comparison of path planning systems for static Obstacles environments in production industries.

Path planning system	20 × 20	50 × 50	100 × 100	20 × 20	50 × 50	100 × 100
CTAs (°)				ATAs (°)		
Classic A*[32]	–	–	–	–	–	–
Safe A*[32]	–	–	–	–	–	–
Improved A*[31]	–	–	–	–	–	–
MAMO + CLS + HS-RNN	77.44	80.77	83.170	12.91	13.46	13.862
PL				PPT		
Classic A*[32]	–	–	–	–	–	–
Safe A*[32]	55.8	–	–	35	–	–
Improved A*[31]	47.6	–	–	26	–	–
MAMO + CLS + HS-RNN	**34.51**	43.61	53.59	3	5	16

Table 2. Results comparison of path planning systems for dynamic Obstacles environments.

0.01 m/s, 1°/s, 0.2 m/s^2, and 50°/s^2. Figures 2 and 3 shows the reproduction results of path planning using planned MAMO + CLS + HS-RNN system for static and dynamic obstacles, respectively.

Comparative analysis of proposed and existing path planning systems

Tables 1 and 2 present a comprehensive comparison of the results gotten from various path preparation systems in motionless static and dynamic obstacle environments. In static and dynamic environment, the proposed MAMO + CLS + HS-RNN system stands out by consistently showing significantly lower PL, CTAs and improved ATAs compared to other systems such as classic A*, safe A*[32], and improved A*[31], respectively. Also, it produced shorter PLs with less path planning time (PPT) in larger grid sizes than other systems. Overall, proposed system outperforms the proportional growth of complexity and erratic pattern-fluctuations.

Path planning system	Accuracy	Precision	Recall	F-Measure	Accuracy	Precision	Recall	F-measure
	Measure (%) – Static obstacles				Measure (%) – Dynamic obstacles			
Random forest	79.714	75.715	75.138	75.426	67.739	66.740	66.399	66.569
Linear regression	82.070	78.071	77.494	77.782	71.301	70.302	69.961	70.131
K-nearest neighbor	84.426	80.427	79.850	80.138	74.863	73.864	73.523	73.693
Support vector machine	86.782	82.783	82.206	82.494	78.425	77.426	77.085	77.255
Decision tree	89.138	85.139	84.562	84.850	81.987	80.988	80.647	80.817
XG boosting	91.494	87.495	86.918	87.206	85.549	84.550	84.209	84.379
Deep neural network	93.850	89.851	89.274	89.562	89.111	88.112	87.771	87.941
Artificial neural network	96.206	92.207	91.630	91.918	92.673	91.674	91.333	91.503
MAMO + CLS + HS-RNN	98.562	94.563	93.986	94.274	96.235	95.236	94.895	95.065

Table 3. Quality comparisons of proposed and existing path planning systems for static and dynamic Obstacles environments.

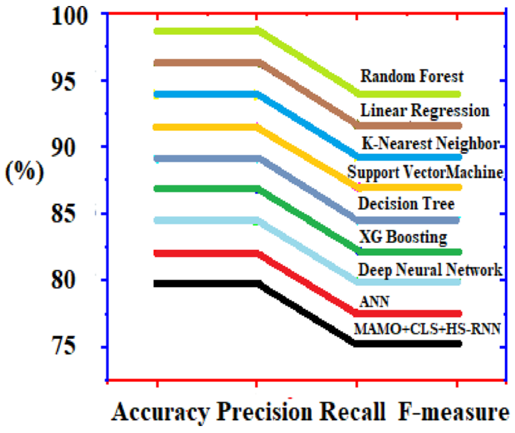


Fig. 4. Quality measure (Static obstacles).

Quality comparison of proposed and existing path planning systems

In this study, each trial's performance metrics, including PL, PPT, CTA, and ATA, were first normalized to allow fair comparison across algorithms. These normalized values were then combined into a single composite performance index (PI) representing the overall efficiency of the path planning process. Success thresholds were defined on the PI to classify trials as optimal or non-optimal, and the MAMO + CLS + HS-RNN algorithm was used as the ground truth to assign labels to all other algorithms. Using these labels, the confusion matrix was calculated to determine Accuracy, Precision, Recall, and F-measure. Finally, the proposed MAMO + CLS + HS-RNN was compared with traditional machine learning algorithms including Random Forest, Linear Regression, K-Nearest Neighbor, Support Vector Machine, Decision Tree, XGBoost, Deep Neural Network, and Artificial Neural Network using these statistical metrics to evaluate its performance and reliability in both static and dynamic path planning scenarios.

Table 3 presents a detailed quality comparison of the proposed MAMO + CLS + HS-RNN system against existing path planning systems for static and dynamic obstacle environments in production industries. In static environment, each system exhibits a progression in measure (%), with the MAMO + CLS + HS-RNN system surpassing them all by achieving improved accuracy, precision, recall and F-measure of 98.562%, 94.563%, 93.986% and 93.986%. Figure 4 shows the system's proficiency in achieving a balance between precision and recall, resulting in accurate and reliable path planning in static obstacle environments. The performance valuation of the MAMO + CLS + HS-RNN and existing models was validated using a confusion matrix, which records the number of correctly and incorrectly classified samples. It consists of four parameters: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These metrics collectively assess the model's classification capability in terms of correctness, sensitivity, and reliability. In the static obstacle environment, 188 out of 200 test samples were correctly classified as positive and 189 as negative, while 12 positives and 11 negatives were misclassified. From these values, Accuracy = (188 + 189) / 200 = 98.5% Precision = 188 / (188 + 11) = 94.56%, Recall = 188 / (188 + 12) = 93.98%, and F-measure = 2 × (0.9456 × 0.9398) / (0.9456 + 0.9398) = 94.27%. These results closely align with the reported performance metrics, validating the model's high classification accuracy and reliability. In dynamic environment, MAMO + CLS + HS-RNN system produced better performance than others by achieving enhanced accuracy, precision, recall, and F-measure of 96.235%, 95.236%, 94.895% and 95.065%. Figure 5 reflects the system's ability to strike a balance between precision and recall, resulting in

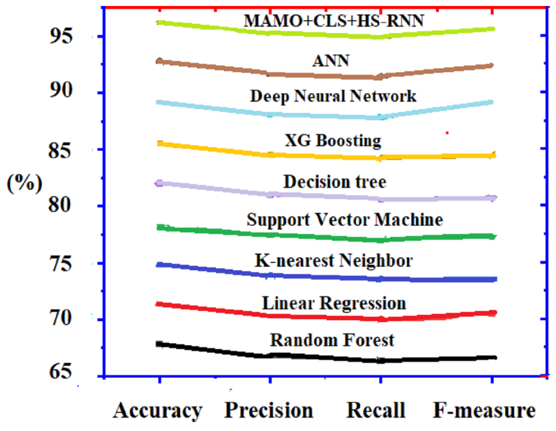


Fig. 5. Quality measure (Dynamic obstacles).

accurate and reliable path planning in dynamic obstacle environment. In test environment, 95 out of 200 tests were correctly predicted as positive and 98 as negative, while 5 positives and 2 negatives were misclassified. From these values, Accuracy = (95 + 98) / 200 = 96.5%, Precision = 95 / (95 + 2) = 95.24%, Recall = 95 / (95 + 5) = 94.90%, and F-measure = 2 × (0.9524 × 0.9490) / (0.9524 + 0.9490) = 95.06%. These computed results closely match the reported values, confirming the model's excellent classification reliability and predictive consistency.

It is revealed from the outcomes that, when the grid size increases, the time also increases to find the optimal results. Though, the MAMO deals through logically examining obstacle states and implementing modifications in the environment. At the same time, when CLS develops chaos theory of finding the optimal path selection is better for the proposed system. The purpose may be the next generation in classic A*, Safe A* and improved A* is strongly affected by this generation's best individuals. This constrains classic A*, Safe A* and improved A* in terms of search region and may have gotten caught in the local optima. Besides the ability of classic A*, Safe A* and improved A* to improve them, they tend to reach prematurely. After that, every iterations of the search region of the proposed system is not strongly controlled through the current best individual. It increases its search region to calculate optimal outcomes and does not get stuck in local minima such as classic A*, Safe A* and improved A*. Further, HS-RNN leverages deep learning to recollect and forecast better paths over time, altering outcomes by new combination learning patterns.

The hybrid framework demonstrates significant advancements over established path planning algorithms, including the hybrid approach presented in Yang and Teng[31]. While such conventional hybrid methods effectively combine improved A∗ planning with local dynamic window approach obstacle avoidance, they typically operate sequentially and can produce suboptimal paths with excessive turns. In contrast, our approach introduces a more sophisticated hierarchical architecture. The multi-population migration component enables parallel exploration through multiple subpopulations, effectively overcoming the premature convergence of standard genetic algorithms. The CLS further refines path quality using chaotic dynamics, providing more comprehensive optimization than the simple path smoothing techniques typically employed in improved A∗ variants. Finally, the HS-RNN performs advanced decision-making for optimal path selection and outperforming the conventional approaches. This integrated methodology generates kinematically superior trajectories with significantly reduced turning angles and enhanced navigation efficiency across diverse environmental conditions.

The proposed algorithm, though effective, has certain limitations. The computation time increases with grid size and environmental complexity, affecting real-time performance. The system's efficiency also depends on careful parameter tuning, and its adaptability may decrease when obstacles move unpredictably. Additionally, as the validation was conducted in a simulated environment, real-time hardware implementation may face processing constraints. Moreover, the HS-RNN model may require retraining for significantly different environments, limiting its generalization capability.

Conclusions

This work, a novel multi-objective optimal navigation system tailored for mobile robots navigating dynamic surroundings, leveraging the effectiveness of hybrid optimization algorithms. The MAMO algorithm is instrumental in computing obstacle state information, enhancing the system's capacity to navigate amidst dynamic obstacles. Additionally, the CLS algorithm is employed for feature extraction and multiple path selection, contributing to the system's adaptability and versatility. The HS-RNN further refines the path planning process by identifying optimal paths and eliminating redundant alternatives. The proposed system overcomes the drawbacks of classic A*, Safe A*, and improved A* through enhancing search regions and removing premature reach. Combining the methods, it guarantees adaptive, multipath, smart optimal path planning even in large grid and dynamic environments. Further, the MAMO + CLS + HS-RNN algorithm was compared with traditional machine learning algorithms using a composite index of normalized path metrics, and performance metrics showed it outperformed all baselines in static and dynamic environments. The confusion matrix based validation confirms the system's high measure (%) in static conditions, and similarly strong performance in dynamic environments, ensuring reliable and precise path planning for industrial applications. The limitations of applied framework include increased computation time increases with larger grid sizes or more complex dynamic environments, affecting real-time performance. Its effectiveness depends on careful parameter tuning, and unpredictable obstacle movements can reduce adaptability. Additionally, real-world implementation may face hardware constraints, and the HS-RNN may require retraining for different environments.

Data availability

We sincerely thank the reviewer for this constructive suggestion. Datasets was generated and/or analyzed during current study available and accessible at web link provided below, as it can effectively handle mobile robot navigation problems, particularly when the outputs involve objectives. "Data availability "The datasets utilized in this research are openly accessible, as they are linked to the paper: https://doi.org/10.1155/2022/2183229. The datasets were generated during and/or analyzed during the current study are available from the corresponding/first author upon reasonable request."Now, it is included in the end Section and relevant reference[31].

Received: 25 August 2025; Accepted: 11 November 2025
Published online: 26 November 2025

References

1. Wu Jie, K., Lin, Jiasen & Sun Pressure or motivation? The effects of low-carbon City pilot policy on china's smart manufacturing. *Comput. Ind. Eng.* **183**, 109512. https://doi.org/10.1016/j.cie.2023.109512 (2023).
2. Zhang, C. et al. Towards new-generation human-centric smart manufacturing in industry 5.0: A systematic review. *Adv. Eng. Inform.* **57**, 102121. https://doi.org/10.1016/j.aei.2023.102121 (2023).
3. Gauttam, H., Pattanaik, K. K. & Bhadauria, S. Garima Nain, and putta Bhanu Prakash, an efficient DNN splitting scheme for edge-AI enabled smart manufacturing. *J. Industrial Inform. Integr.*. **34**, 100481. https://doi.org/10.1016/j.jii.2023.100481 (2023).
4. Bhatia, P. & Diaz-Elsayed, N. Facilitating decision-making for the adoption of smart manufacturing technologies by SMEs. *Via Fuzzy TOPSIS Int. J. Prod. Econ.*. **257**, 108762. https://doi.org/10.1016/j.ijpe.2022.108762 (2023).
5. Li, C., Zheng, P., Yin, Y., Wang, B. & Wang, L. Deep reinforcement learning in smart manufacturing: A review and prospects. *CIRP J. Manufact. Sci. Technol.* **40**, 75–101. https://doi.org/10.1016/j.cirpj.2022.11.003 (2023).
6. De Giacomo, G., Favorito, M., Leotta, F., Mecella, M. & Silo, L. Digital twin composition in smart manufacturing via Markov decision processes. *Comput. Industry*, **149**, 1–10. https://doi.org/10.1016/j.compind.2023.103916 (2023).
7. Feng, Q. et al. Multi-level predictive maintenance of smart manufacturing systems driven by digital twin: A matheuristics approach. *J. Manuf. Syst.*, **68**,. https://doi.org/10.1016/j.jmsy.2023.05.004
8. Eyuboglu, M. & Atali, G. A novel collaborative path planning algorithm for 3-wheel omnidirectional autonomous mobile Robot,Robotics Autonomous Syst., **169**, 104527. https://doi.org/10.1016/j.robot.2023.104527 (2023).
9. Mısır, O. Dynamic local path planning method based on neutrosophic set theory for a mobile robot. *J. Brazilian Soc. Mech. Sci. Eng.* **45** (3), 127. https://doi.org/10.1007/s40430-023-04048-6 (2023).
10. Kumar, S., Dadas, S. S. & Parhi, D. R. Path planning of mobile robot using modified DAYKUN-BIP virtual target displacement method in static environments, *Wireless Personal Commun.*, **128** (3), 2287–2305. https://doi.org/10.1007/s11277-022-10043-2 (2023).
11. Dai, J., Zhang, Y. & Deng, H. Novel potential guided bidirectional RRT* with direct connection strategy for path planning of redundant robot manipulators in joint space. *IEEE Trans. Industr. Electron.* **71** (3), 2737–2747. https://doi.org/10.1109/TIE.2023.3269462 (2023).
12. Zhang, R., Chai, R., Chai, S., Xia, Y. & Tsourdos, A. Design and practical implementation of a high efficiency two-layer trajectory planning method for AGV, *IEEE Trans. Industrial Electron.*, **71**(2), 1811–1822.https://doi.org/10.1109/TIE.2023.3250847 (2023).
13. Zhao, L. et al. Graph-based robust localization of object-level map for mobile robotic navigation. *IEEE Trans. Industrial Electron.*. **71** (1), 697–707. https://doi.org/10.1109/TIE.2023.3245208 (2023).
14. Zhang, X., Ji, Y., Wang, C., Lin, H. & Wang, Y. Path planning of inspection robot based on improved intelligent water drop algorithm. *IEEE Access.* **11**, 119993–120000. https://doi.org/10.1109/ACCESS.2023.3326756 (2023).
15. Yang, H., Yao, C., Liu, C. & Chen, Q. Rmrl: robot navigation in crowd environments with risk map-based deep reinforcement learning. *IEEE Rob. Autom. Lett.*. **8** (12), 7930–7937. https://doi.org/10.1109/LRA.2023.3322093 (2023).
16. Li, Yong & Cheng, H. A* post-processing algorithm for robots with bidirectional shortcut and path perturbation. *IEEE Rob. Autom. Lett.* **8** (11), 7775–7782. https://doi.org/10.1109/LRA.2023.3320432 (2023).
17. Dhouib, S. Shortest path planning via the rapid Dhouib-Matrix-SPP (DM-SPP) method for the autonomous mobile robot, results in control and optimization, **13**, 100299. https://doi.org/10.1016/j.rico.2023.100299 (2023).
18. Ou Junlin, S. H., Hong, G., Song & Wang, Y. Hybrid path planning based on adaptive visibility graph initialization and edge computing for mobile robots. *Eng. Appl. Artif. Intell. Volume*. **126** (1), 107110. https://doi.org/10.1016/j.engappai.2023.107110 (2023).
19. Yu, Z. et al. A path planning algorithm for mobile robot based on water flow potential field method and beetle antennae search algorithm, *Comput. Electrical Eng.*, **109**, https://doi.org/10.1016/j.compeleceng.2023.108730 (2023).
20. Bulut, V. Path planning of mobile robots in dynamic environment based on analytic geometry and cubic Bézier curve with three shape parameters. *Expert Syst. Appl.*. **233** (3), 120942. https://doi.org/10.1016/j.eswa.2023.120942 (2023).
21. Luo, Q., Wang, H., Zheng, Y. & He, J. Research on path planning of mobile robot based on improved ant colony algorithm. *Neural Comput. Appl.*, **32**, 1555–1566. https://doi.org/10.1007/s00521-019-04172-2 (2020).
22. Maoudj, A. & Hentout, A. Optimal path planning approach based on Q-learning algorithm for mobile robots. *Appl. Soft Comput.* **97**, 106796. https://doi.org/10.1016/j.asoc.2020.106796 (2020).
23. Ajeil, F. H., Kasim, I., Ibraheem, M. A., Sahib & Humaidi, A. J. Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. *Appl. Soft Comput.* **89**, 106076. https://doi.org/10.1016/j.asoc.2020.106076 (2020).
24. Chen, Y. et al. Autonomous mobile robot path planning in unknown dynamic environments using neural dynamics, *Soft comput.*, **24** (18), 13979–13995. https://doi.org/10.1007/s00500-020-04771-5 (2020).
25. Sun, H. et al. An improved path planning algorithm for mobile robots. *J. Syst. Simul.*. **36** (9), 2193–2207. https://doi.org/10.16182/j.issn1004731x.joss.23-0585 (2024).
26. Miao, C., Chen, G., Wu, Y. & Chengliang Yan, and Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. *Comput. Ind. Eng.* **156**, 107230. https://doi.org/10.1016/j.cie.2021.107230 (2021).
27. Song, B., Wang, Z. & Zou, L. An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve, *Appl. Soft Comput.*, **100**, https://doi.org/10.1016/j.asoc.2020.106960 (2021).
28. Lyu, D., Chen, Z., Cai, Z. & Piao, S. Robot path planning by leveraging the graph-encoded Floyd algorithm, *Future Generation Comput. Syst.*, **122** https://doi.org/10.1016/j.future.2021.03.007, (2021).
29. Zhang, Z., Qiao, B., Zhao, W. & Chen, X. A predictive path planning algorithm for mobile robot in dynamic environments based on rapidly exploring random tree. *Arab. J. Sci. Eng.* **46** https://doi.org/10.1007/s13369-021-05443-8 (2021).

30. Zafar, M., Nayab, J. C., Mohanta & Keshari, A. GWO-potential field method for mobile robot path planning and navigation control. *Arab. J. Sci. Eng.* **46** https://doi.org/10.1007/s13369-021-05487-w (2021).
31. Yang, H. & Teng, X. Mobile robot path planning based on enhanced dynamic window approach and improved A∗ algorithm, *J. Robot.*, **2022** https://doi.org/10.1155/2022/2183229 (2022).
32. Zhong, X., Tian, J. & Peng, X. Huosheng Hu, and Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment. *J. Intell. Robotic Syst.* **99** (1), 65–77. (2024).
33. Raj, R. & Kos, A. Intelligent mobile robot navigation in unknown and complex environment using reinforcement learning technique, *Sci. Rep.*, **14** (1), 22852. (2024).
34. Luo, Q., Ma, M. & Zhou, Y. A novel animal migration algorithm for global numerical optimization. *Comput. Sci. Inf. Syst.*. **13** (1), 259–285. https://doi.org/10.2298/CSIS141229041L (2016).
35. Camarena, O., Cuevas, E., Pérez-Cisneros, M. & Fausto, F. Adrián González, and Arturo Valdivia, Ls-II: an improved locust search algorithm for solving optimization problems, *Mathem. Probl. Eng.*, **1**, 1–15. https://doi.org/10.1155/2018/4148975 (2018).
36. Tunay, M. & Abiyev, R. Improved hypercube optimisation search algorithm for optimisation of high dimensional functions. *Math. Probl. Eng.* (1), 1–13. https://doi.org/10.1155/2022/6872162 (2022).
37. Deo, D. R. et al. Brain control of bimanual movement enabled by recurrent neural networks, *Sci. Rrep.*, **14** (1), 1598. (2024). s41598-024-51617-3.
38. Hao, K., Zhao, J., Yu, K., Li, C. & Wang, C. Path planning of mobile robots based on a multi-population migration genetic algorithm, *Sensors*, **20**, https://doi.org/10.3390/s20205873 (2020).

Acknowledgements
The authors thank Dr. S. Ganesh Kumar (Conception et optimisation de problèmes de disposition utilisant des métaheuristiques, Scientia Iranica) for the assistance in the mobile robot navigation paper.

Author contributions
K. designed the research framework and developed the multi-objective optimal navigation system. K. implemented the Modified Animal's Migration Optimization (MAMO) algorithm for obstacle state analysis and proactive collision avoidance. K. further integrated the Chaos Locust Search (CLS) and Hypercube Search with Recurrent Neural Networks (HS-RNN) for multi-path generation and optimal path refinement. K. carried out the simulations, analyzed the results, and prepared the manuscript. All aspects of the study were conducted by K and reviewed by S.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to K.R.

Reprints and permissions information is available at www.nature.com/reprints.